# **Grid Navigation Report**

**Team Members**: Trevor Davenport, Phuoc Nguyen, Khoa Tran, Corey Short

**Brief Description**: Khoa and Phuoc worked collectively to develop the java coding while Corey and Trevor collected data and helped implement algorithm design.

**Approximate Number of Hours**: 20-25 Hours.

**Project Repository Location**: <https://github.com/IEOR140-T5/Project2NXT/>

**Performance Specifications We Met**: Both Basic and Extra Credit.

**Hardware Design**: Our hardware design was the exact same implementation as last project, with the exception of adding and calibrating the light sensors.

**Experimental Work**: We have done much experimental work, for example, we calculated the TurnRate and the center of the line.

* **Experiment Description and Purpose**: The purpose of this project was to demonstrate our usage of the leJOS java doc along with modifying the engineering behind our Rover.
* **Listing of Data**: Attached within our repository are the three data graphs we used to calibrate our Rover's light senors.
* **Calculations and Analysis**: After some trial and error with control values, we found that the value 1.05 for the turnrate was superior in allowing our rover to remain steady.
* **Implementations in Code**: As stated above, we used the control variable and set it to 1.05.

**Problem Analysis**

* **Subtasks**
  + **Data and Algorithms**: Our algorithm for computing the turnrate involved using a method provided ( CLDistance ) to determine the distance between each sensor and how far off the desired path our rover was. Next, after calculating this distance, we implemented the Steer built-in method from lejos with a control variable set to 1.02. The next algorithm we developed for Milestone 2 goes as follows:

1. Loop Necessary Iterations

2. Determine if the Sensors have come in contact with a black marker

3. If true, travel X distance and Sleep Y ms.

4. Next, Determine if the current iteration is odd.

5. If true, Sleep Y ms, TurnRobot X direction, and then Sleep Y again.

6. Else, continue to trackLine().

* + **Task Sequence and Repetition**: After running multiple tests with the oval track, we were able to determine correct values while observing how our rover reacted to changes in code.

**Software Design**

* **Sources of Programming Ideas**: we use the while loop to keep calling the method trackLine() to determine the center of the line and to steer at a desired turn rate in order to keep on track. Beside, we count the number of black marker that the robot has passed, the travel a small distance in order for the robot for not detect the black market anymore. The robot will stop or turn when it reaches a specific numbers of black market detections.
* **Class Responsibilities**: The class Tracker take responsiblity for the calibration setup and basic control of the robot. For Milestone 2, the only method we implement is go() which is detailed above.
* **Other classes used**: None
* **Explain how submethods work**: It help the tracker and the robot to stop, delay, or turn at a desired angle.
* **Link to Documentation**: Team5/Project2/doc

**Most Interesting/Challenging Parts**: We found that the most interesting portion of the project was implementing the light sensors. The light sensors, in our opinion, were the most intriguing aspect of the project. After figuring out how to correctly calibrate their values, the coding and algorithms came next. We found that developing the algorithm and implementing code was the most difficult portion of this project.

**Link to source code**: Team5/Project2/

## **Milestone 1**

## In this milestone, we tried to calculate the center of the line, the turn rate to make the robot not losing its track. For the main part, the robot will trace the line oval track 4 times, turn back and do more 4 rounds. For the extra credit part, it will trace the oval track but in a way of 8.

## **Milestone 2**

## In Milestone 2, our group used a method named go(). This method begins by calibrating the tracker and then runs our algorithm designed by Peter. Next, the method is seperated into four seperate for loops (2 rounds of left and right turns and next uses coordinate mappings). Based on our light sensor data,we deduced that if the left or right values ever read less than -10, we know that the rover has come across the black marking. We than pilot the rover, put it to sleep, and as our iterations for each for loop are odd, the robot will turn, sleep, and then continue to trackLine(). This same basic algorithm is implemented throughout our go() method.